回文自动机（姬）

算法思想

回文自动机是一种可以存储一个串中所有回文子串的数据结构。

与其他自动机类似，回文自动机也有转移边和失配指针 $fail$ 组成，每个结点都代表一个回文子串。

对于奇数和偶数的回文串，我们分别建立一棵树。一个节点的 $fail$ 指针指向的是这个节点所代表的回文串的最长回文后缀所对应的节点。转移边代表在原结点代表的回文串前后各加一个相同的字符。还需要对每个结点维护回文子串的长度 len 。

构造自动机

两个初始状态分别代表 $-1,0$ 的回文串，分别叫做奇根和偶根。偶根的 $fail$ 指针指向奇根，我们不用管奇根的 $fail$ 指针，因为奇根不可能失配（因为最短的奇数回文串是单个字符串，这显然存在，不可能失配）。

插入字符

现在假设已经构造完前 $p-1$ 个字符的回文自动机后，现在插入位置为 p 的字符。

从上一个字符结尾的最长回文子串对应的结点开始，一直沿着 $fail$ 走，直到找到一个结点满足 $s_{p}=s_{p-len-1}$ 就说明这个位置和这个回文串之前的一个字符一样，所以可以构成新的回文串。如果到最后都没找到 len 为 -1，正好是 $s_{p}=s_{p}$ 说明没有这个结点，需要特判新建。

我们还需要求出新建的结点的 $fail$ 指针，具体方法和上面的过程类似，不断跳转 $fail$ 指针，从不加新字符的那个回文串开始，就可以找到另一个带着两个新字符的回文串，然后把 $fail$ 指针指到这个字符串即可。如果 $fail$ 没匹配到，那么将它连向长度为 0 的那个结点，显然可以（因为这是所有节点的后缀）。

对于一个字符串 s 它的本质不同的回文子串个数最多只有 $|s|$ 个（数学归纳法）。所以转移状态数也是 $O(|s|)$ 的。

算法实现

```
#include <bits/stdc++.h>
using namespace std;
const int maxn=1001000;
struct PAM {
    char s[maxn];
    int len[maxn],n,num[maxn],fail[maxn],last,cur,pos,trie[maxn][26],tot;
    void init() {
        memset(s,0,sizeof(s));
        memset(len,0,sizeof(len));
        memset(num,0,sizeof(num));
```

```
        memset(fail,0,sizeof(fail));
        memset(trie,0,sizeof(trie));
        n=0;last=0;cur=0;pos=0;tot=1;
        fail[0]=1;len[1]=-1;
    }
    int getfail(int x,int i) {
        while(i-len[x]-1<0||s[i-len[x]-1]!=s[i])x=fail[x];
        return x;
    }
    void solve() {
        for(int i=0; i<=n-1; i++) {
            pos=getfail(cur,i);
            if(!trie[pos][s[i]-'a']) {
                fail[++tot]=trie[getfail(fail[pos],i)][s[i]-'a'];
                trie[pos][s[i]-'a']=tot;
                len[tot]=len[pos]+2;
                num[tot]=num[fail[tot]]+1;
            }
            cur=trie[pos][s[i]-'a'];
            last=num[cur];
            printf("%d ",last);
        }
    }
} p;

int main() {
    p.init();
    scanf("%s",p.s);
    p.n=strlen(p.s);
    p.solve();
    return 0;
}
```